RESEARCH CENTRE

Sophia Antipolis - Méditerranée

2020 ACTIVITY REPORT

Project-Team STAMP

Safety Techniques based on Formalized Mathematical Proofs

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Contents

Project-Team STAMP

Creation of the Project-Team: 2019 November 01

Keywords

Computer sciences and digital sciences

A2.1.11. – Proof languages

A2.4.3. – Proofs

A4.5. – Formal methods for security

A5.10.3. – Planning

A7.2. – Logic in Computer Science

A7.2.3. – Interactive Theorem Proving

A7.2.4. – Mechanized Formalization of Mathematics

A8.3. – Geometry, Topology

A8.4. – Computer Algebra

A8.10. – Computer arithmetic

Other research topics and application domains

B6.1. – Software industry

B9.5.1. – Computer science

B9.5.2. – Mathematics

1 Team members, visitors, external collaborators

Research Scientists

- Yves Bertot [Team leader, Inria, Senior Researcher, HDR]
- Cyril Cohen [Inria, Researcher]
- Benjamin Grégoire [Inria, Researcher]
- Laurence Rideau [Inria, Researcher]
- Enrico Tassi [Inria, Researcher]
- Laurent Théry [Inria, Researcher]

Post-Doctoral Fellows

- Pierre Boutry [Inria]
- Christian Doczkal [Univ Côte d'Azur]
- Jean Christophe Lechenet [Inria, from Oct 2020]

PhD Students

- Cécile Baritel-Ruet [Inria, until Jul 2020]
- Mohamad El Laz [Inria, until Nov 2020]
- Swarn Priya [Inria, from Oct 2020]

Technical Staff

- Maxime Dénès [Inria, Engineer]

Interns and Apprentices

- Come Neyrand [École normale supérieure de Rennes, from Jun 2020 until Jul 2020]

Administrative Assistant

- Nathalie Bellesso [Inria]

Visiting Scientists

- Reynald Affeldt [National Institute of Advanced Industrial Science and Technology-Japan, until Jul 2020]
- Swarn Priya [Purdue University - USA, until Apr 2020]

External Collaborators

- Gilles Barthe [Max Planck Institute SP, HDR]
- Loïc Pottier [Ministère de l'Education Nationale, HDR]

2 Overall objectives

Computers and programs running on these computers are powerful tools for many domains of human activities. In some of these domains, program errors can have enormous consequences. It will become crucial for all stakeholders that the best techniques are used when designing these programs.

We advocate using higher-order logic proof assistants as tools to obtain better quality programs and designs. These tools make it possible to build designs where all decisive arguments are explicit, ambiguity is alleviated, and logical steps can be verified precisely. In practice, we are intensive users of the Coq system and we participate actively to the development of this tool, in collaboration with other teams at Inria, and we also take an active part in advocating its usage by academic and industrial users around the world.

Many domains of modern computer science and engineering make a heavy use of mathematics. If we wish to use proof assistants to avoid errors in designs, we need to develop corpora of formally verified mathematics that are adapted to these domains. Developing libraries of formally verified mathematics is the main motivation for our research. In these libraries, we wish to capture not only the knowledge that is usualy recorded in definitions and theorems, but also the practical knowledge that is recorded in mathematical practice, idioms, and work habits. Thus, we are interested in logical facts, algorithms, and notation habits. Also, the very process of developing an ambitious library is a matter of organisation, with design decisions that need to be evaluated and improved. Refactoring of libraries is also an important topic. Among all higher-order logic based proof assistants, we contend that those based on Type theory are the best suited for this work on libraries, thanks to their strong capabilities for abstraction and modular re-use.

The interface between mathematics, computer science and engineering is large. To focus our activities, we will concentrate on applications of proof assistants to two main domains: cryptography and robotics. We also develop specific tools for proofs in cryptography, mainly around a proof tool named EasyCrypt.

3 Research program

3.1 Theoretical background

The proof assistants that we consider provide both a programming language, where users can describe algorithms performing tasks in their domain of interest, and a logical language to reason about the programs, thus making it possible to ensure that the algorithms do solve the problems for which they were designed. Trustability is gained because algorithms and logical statements provide multiple views of the same topic, thus making it possible to detect errors coming from a mismatch between expected and established properties. The verification process is itself a logical process, where the computer can bring rigor in aligning expectations and guarantees.

The foundations of proof assistants rest on the very foundations of mathematics. As a consequence, all aspects of reasoning must be made completely explicit in the process of formally verifying an algorithm. All aspects of the formal verification of an algorithm are expressed in a discourse whose consistency is verified by the computer, so that unclear or intuitive arguments need to be replaced by precise logical inferences.

One of the foundational features on which we rely extensively is *Type Theory*. In this approach a very simple programming language is equiped with a powerful discipline to check the consistency of usage: types represent sets of data with similar behavior, functions represent algorithms mapping types to other types, and the consistency can be verified by a simple computer program, a *type-checker*. Although they can be verified by a simple program, types can express arbitrary complex objects or properties, so that the verification work lives in an interesting realm, where verifying proofs is decidable, but finding the proofs is undecidable.

This process for producing new algorithms and theorems is a novelty in the development of mathematical knowledge or algorithms, and new working methods must be devised for it to become a productive approach to high quality software development. Questions that arise are numerous. How do we avoid requiring human assistance to work on mundane aspects of proofs? How do we take advantage of all the progress made in automatic theorem proving? How do we organize the maintenance of ambitious corpora of formally verified knowledge in the long term?

To acquire hands-on expertise, we concentrate our activity on three aspects. The first one is foundational: we develop and maintain a library of mathematical facts that covers many aspects of algebra. In the past, we applied this library to proofs in group theory, but it is increasingly used for many different areas of mathematics and by other teams around the world, from combinatorics to elliptic cryptography, for instance. The second aspect is applicative: we develop a specific tool for proofs in cryptography, where we need to reason on the probability that opponents manage to access information we wish to protect. For this activity, we develop a specific proof system, relying on a wider set of automatic tools, with the objective of finding the tools that are well adapted to this domain and to attract users that are initially specialists in cryptography but not in formal verification. The third domain is robotics, as we believe that the current trend towards more and more autonomous robots and vehicles will raise questions of safety and trustability where formal verification can bring significant added value.

4 Application domains

4.1 Mathematical Components

The Mathematical Components is the main by-product of an effort started almost two decades ago to provide a formally verified proof for a major theorem in group theory. Because this major theorem had a proof published in books of several hundreds of pages, with elements coming from character theory, other coming from algebra, and some coming from real analysis, it was an exercice in building a large library, with results in many domains, and in establishing clear guidelines for further increase and data search.

This library has proved to be a useful repository of mathematical facts for a wide area of applications, so that it has a growing community of users in many contries (Denmark, France, Germany, Japan, Singapore, Spain, Sweden, UK, USA) and for a wide variety of topics (transcendental number theory, elliptic curve cryptography, articulated robot kinematics, recently block chain foundations).

Interesting questions on this library range around the importance of decidability and proof irrelevance, the way to structure knowledge to automatically inherit theorems from one topic to another, the way to generate infrastructure to make this automation efficient and predictable. In particular, we want to concentrate on adding a new mathematical topic to this library: real analysis and then complex analysis (Mathematical Components Analysis).

On the front of automation, we are convinced that a higher level language is required to describe similarities between theories, to generate theorems that are immediate consequences of structures, etc, and for this reason, we invest in the development of a new language on top of the proof assistant (ELPI).

4.2 Proofs in cryptography

When we work on cryptography, we are interested in the formal verification of proofs showing that some cryptographic primitives provide good guarantees against unwanted access to information. Over the years we have developed a technique for this kind of reasoning that relies on a programing logic (close to Hoare logic) with probabilistic aspects and the capability to establish relations between several implementations of a problem. The resulting programming logic is called *probabilistic relational Hoare logic*. In more recent work, we have also started to study questions of *side-channel* attacks, where we wish to guarantee that opponents cannot gain access to protected knowledge, even if they observe specific features of execution, like execution time (to which the answer lies in *constant-time* execution) or partial access to memory bits (to which the answer lies in *masking*).

For this domain of application, we choose to work with a specific proof tool (EasyCrypt), which combines powerful first-order reasoning and uses of automatic tools, with a specific support for probabilistic relational Hoare Logic. The development of this EasyCrypt proof tool is one of the objectives of our team.

When it comes to formal proofs of resistance to side-channel attack, we contend that it is necessary to verify formally that the compiler used in the production of actually running code respects the resistance properties that were established in formally verified proofs. One of our objectives is to describe such a compiler (Jasmin) and show its strength on a variety of applications.

4.3 Proofs for robotics

Robots are man-made artifacts where numerous design decisions can be argued based on logical or mathematical principles. For this reason, we wish to use this domain of application as a focus for our investigations. The questions for which we are close to providing answers involve precision issues in numeric computation, obstacle avoidance and motion planning (including questions of graph theory), articulated limb cinematics and dynamics, and balance and active control.

From the mathematical perspective, these topics require that we improve our library to cover real algebraic geometry, computational geometry, real analysis, graph theory, and refinement relations between abstract algorithms and executable programs.

In the long run, we hope to exhibit robots where pieces of software and part of the design has been subject to formal verification.

5 New software and platforms

5.1 New software

5.1.1 Coq

Name: The Coq Proof Assistant

Keywords: Proof, Certification, Formalisation

Scientific Description: Coq is an interactive proof assistant based on the Calculus of (Co-)Inductive Constructions, extended with universe polymorphism. This type theory features inductive and co-inductive families, an impredicative sort and a hierarchy of predicative universes, making it a very expressive logic. The calculus allows to formalize both general mathematics and computer programs, ranging from theories of finite structures to abstract algebra and categories to programming language metatheory and compiler verification. Coq is organised as a (relatively small) kernel including efficient conversion tests on which are built a set of higher-level layers: a powerful proof engine and unification algorithm, various tactics/decision procedures, a transactional document model and, at the very top an IDE.

Functional Description: Coq provides both a dependently-typed functional programming language and a logical formalism, which, altogether, support the formalisation of mathematical theories and the specification and certification of properties of programs. Coq also provides a large and extensible set of automatic or semi-automatic proof methods. Coq's programs are extractible to OCaml, Haskell, Scheme, ...

Release Contributions: Some highlights from this release are:

- Introduction of primitive persistent arrays in the core language, implemented using imperative persistent arrays.
- Introduction of definitional proof irrelevance for the equality type defined in the SProp sort.
- Many improvements to the handling of notations, including number notations, recursive notations and notations with bindings. A new algorithm chooses the most precise notation available to print an expression, which might introduce changes in printing behavior.

See the Zenodo citation https://zenodo.org/record/4501022#.YB00r5NKj1w for more information on this release.

News of the Year: Coq version 8.13 integrates many usability improvements, as well as extensions of the core language. The main changes include:

- Introduction of primitive persistent arrays in the core language, implemented using imperative persistent arrays.
- Introduction of definitional proof irrelevance for the equality type defined in the SProp sort.

- Cumulative record and inductive type declarations can now specify the variance of their universes.
- Various bugfixes and uniformization of behavior with respect to the use of implicit arguments and the handling of existential variables in declarations, unification and tactics.
- New warning for unused variables in catch-all match branches that match multiple distinct patterns.
- New warning for Hint commands outside sections without a locality attribute, whose goal is to eventually remove the fragile default behavior of importing hints only when using Require. The recommended fix is to declare hints as export, instead of the current default global, meaning that they are imported through Require Import only, not Require. See the following rationale and guidelines for details.
- General support for boolean attributes.
- Many improvements to the handling of notations, including number notations, recursive notations and notations with bindings. A new algorithm chooses the most precise notation available to print an expression, which might introduce changes in printing behavior.
- Tactic improvements in lia and its zify preprocessing step, now supporting reasoning on boolean operators such as Z.leb and supporting primitive integers Int63.
- Typing flags can now be specified per-constant / inductive.
- Improvements to the reference manual including updated syntax descriptions that match Coq's grammar in several chapters, and splitting parts of the tactics chapter to independent sections.

See the changelog for an overview of the new features and changes, along with the full list of contributors. https://coq.github.io/doc/v8.13/refman/changes.html#version-8-13

URL: http://coq.inria.fr/

Authors: Bruno Barras, Yves Bertot, Frédéric Besson, Pierre Corbineau, Cristina Cornes, Judicaël Courant, Pierre Courtieu, Pierre Crégut, David Delahaye, Maxime Denes, Jean-Christophe Filliâtre, Julien Forest, Emilio Jesus Gallego Arias, Gaëtan Gilbert, Georges Gonthier, Benjamin Grégoire, Hugo Herbelin, Gérard Huet, Vincent Laporte, Pierre Letouzey, Assia Mahboubi, Pascal Manoury, Guillaume Melquiond, César Munoz, Chetan Murthy, Amokrane Saibi, Catherine Parent, Christine Paulin Mohring, Pierre-Marie Pédrot, Loïc Pottier, Matthieu Sozeau, Arnaud Spiwack, Enrico Tassi, Laurent Théry, Benjamin Werner, Théo Zimmermann

Contacts: Hugo Herbelin, Matthieu Sozeau

Participants: Yves Bertot, Frédéric Besson, Tej Chajed, Cyril Cohen, Pierre Corbineau, Pierre Courtieu, Maxime Denes, Jim Fehrle, Julien Forest, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Georges Gonthier, Benjamin Grégoire, Jason Gross, Hugo Herbelin, Vincent Laporte, Olivier Laurent, Assia Mahboubi, Kenji Maillard, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, Clément Pit-Claudel, Kazuhiko Sakaguchi, Vincent Semeria, Michael Soegtrop, Arnaud Spiwack, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Anton Trunov, Li-Yao Xia, Théo Zimmermann

Partners: CNRS, Université Paris-Sud, ENS Lyon, Université Paris-Diderot

5.1.2 Math-Components

Name: Mathematical Components library

Keyword: Proof assistant

Functional Description: The Mathematical Components library is a set of Coq libraries that cover the prerequiste for the mechanization of the proof of the Odd Order Theorem.

Release Contributions: This release is compatible with Coq 8.10, 8.11 and Coq 8.12. The main changes are:

- support for Coq 8.7, 8.8 and 8.9 have been dropped,

- a change of implementation of intervals and the updated theory,

- the addition of kernel lemmas for matrices,

- generalized many lemmas for path and sorted,

- several lemma additions, name changes and bug fixes.

URL: http://math-comp.github.io/math-comp/

Contacts: Assia Mahboubi, Enrico Tassi, Georges Gonthier

Participants: Alexey Solovyev, Andrea Asperti, Assia Mahboubi, Cyril Cohen, Enrico Tassi, François Garillot, Georges Gonthier, Ioana Pasca, Jeremy Avigad, Laurence Rideau, Laurent Théry, Russell O'Connor, Sidi Ould Biha, Stéphane Le Roux, Yves Bertot

5.1.3 Easycrypt

Keywords: Proof assistant, Cryptography

Functional Description: EasyCrypt is a toolset for reasoning about relational properties of probabilistic computations with adversarial code. Its main application is the construction and verification of game-based cryptographic proofs. EasyCrypt can also be used for reasoning about differential privacy.

URL: https://www.easycrypt.info/trac/

Contact: Gilles Barthe

Participants: Benjamin Grégoire, Gilles Barthe, Pierre-Yves Strub

5.1.4 ELPI

Name: Embeddable Lambda Prolog Interpreter

Keywords: Constraint Programming, Programming language, Higher-order logic

Scientific Description: The programming language has the following features

- Native support for variable binding and substitution, via an Higher Order Abstract Syntax (HOAS) embedding of the object language. The programmer needs not to care about De Bruijn indexes.

- Native support for hypothetical context. When moving under a binder one can attach to the bound variable extra information that is collected when the variable gets out of scope. For example when writing a type-checker the programmer needs not to care about managing the typing context.

- Native support for higher-order unification variables, again via HOAS. Unification variables of the meta-language (lambdaProlog) can be reused to represent the unification variables of the object language. The programmer does not need to care about the unification-variable assignment map and cannot assign to a unification variable a term containing variables out of scope, or build a circular assignment.

- Native support for syntactic constraints and their meta-level handling rules. The generative semantics of Prolog can be disabled by turning a goal into a syntactic constraint (suspended goal). A syntactic constraint is resumed as soon as relevant variables get assigned. Syntactic constraints can be manipulated by constraint handling rules (CHR).

- Native support for backtracking, to ease implementation of search.

- The constraint store is extensible. The host application can declare non-syntactic constraints and uses custom constraint solvers to check their consistency.

- Clauses are graftable. The user is free to extend an existing program by inserting/removing clauses, both at runtime (using implication) and at "compilation" time by accumulating files.

Most of these features come with lambdaProlog. Constraints and propagation rules are novel in ELPI.

Functional Description: ELPI implements a variant of lambdaProlog enriched with Constraint Handling Rules, a programming language well suited to manipulate syntax trees with binders and unification variables.

ELPI is a research project aimed at providing a programming platform for the so called elaborator component of an interactive theorem prover.

ELPI is designed to be embedded into larger applications written in OCaml as an extension language. It comes with an API to drive the interpreter and with an FFI for defining built-in predicates and data types, as well as quotations and similar goodies that come in handy to adapt the language to the host application.

Release Contributions: - Improvements to the parser (parsing negative numbers)

- Improvements to the foreign function interface (accepting ternary comparison, instead of equality)

- addition of ternary comparisons to the standard library

- inclusion of a builtin comparison cmp_term

- inclusion of a builtin function to check whether a term is ground

URL: https://github.com/lpcic/elpi/

Publications: hal-01176856, hal-01410567, hal-01897468

Contact: Enrico Tassi

Participants: Claudio Sacerdoti Coen, Enrico Tassi

5.1.5 coq-elpi

Keywords: Metaprogramming, Extension

Scientific Description: Coq-elpi provides a Coq plugin that embeds ELPI. It also provides a way to embed Coq's terms into lambdaProlog using the Higher-Order Abstract Syntax approach (HOAS) and a way to read terms back. In addition to that it exports to ELPI a set of Coq's primitives, e.g. printing a message, accessing the environment of theorems and data types, defining a new constant and so on. For convenience it also provides a quotation and anti-quotation for Coq's syntax in lambdaProlog. E.g. {{nat}} is expanded to the type name of natural numbers, or {{A -> B}} to the representation of a product by unfolding the -> notation. Finally it provides a way to define new vernacular commands and new tactics.

Functional Description: Coq plugin embedding ELPI

Release Contributions: Minor relase for extra API for global reference data types

Publications: hal-01897468, hal-01637063

Contact: Enrico Tassi

Participant: Enrico Tassi

5.1.6 MaskComp

Keyword: Masking

Functional Description: MaskComp is a compiler generating masked implementations protected against side channel attacks based on differential power analysis. It takes a unmasked program in a syntax close to C and generates a new C protected program. We do not claim that the generated C program will be secure after compilation (the C compiler may break protection), but it provides a good support for generating masked implementation.

URL: https://sites.google.com/site/maskingcompiler/home

Contact: Benjamin Grégoire

5.1.7 Jasmin

Name: Jasmin compiler and analyser

Keywords: Cryptography, Static analysis, Compilers

Functional Description: The Jasmin programming language smoothly combines high-level and low-level constructs, so as to support "assembly in the head" programming. Programmers can control many low-level details that are performance-critical: instruction selection and scheduling, what registers to spill and when, etc. The language also features high-level abstractions (variables, functions, arrays, loops, etc.) to structure the source code and make it more amenable to formal verification. The Jasmin compiler produces predictable assembly and ensures that the use of high-level abstractions incurs no run-time penalty.

The semantics is formally defined to allow rigorous reasoning about program behaviors. The compiler is formally verified for correctness (the proof is machine-checked by the Coq proof assistant). This justifies that many properties can be proved on a source program and still apply to the corresponding assembly program: safety, termination, functional correctness...

Jasmin programs can be automatically checked for safety and termination (using a trusted static analyzer). The Jasmin workbench leverages the EasyCrypt toolset for formal verification. Jasmin programs can be extracted to corresponding EasyCrypt programs to prove functional correctness, cryptographic security, or security against side-channel attacks (constant-time).

URL: https://github.com/jasmin-lang/jasmin

Publication: hal-02974993

Contacts: Benjamin Grégoire, Vincent Laporte, Adrien Koutsos

Participants: Gilles Barthe, Benjamin Grégoire, Adrien Koutsos, Vincent Laporte

5.1.8 MaskVerif

Name: MaskVerif

Keywords: Masking, Hardware and Software Platform

Functional Description: MaskVerif is a tool to verify the security of implementations protected against side channel attacks, in particular differential power analysis. It allows to check different security notions in the probing model: - Probing security - Non Interference - Strong Non Interference. The tool is able to analyse software implementations and hardware implementations (written in Verilog). It can prove the different security notions in presence of glitch or transition.

URL: https://sites.google.com/view/maskverif/home

Contact: Benjamin Grégoire

5.1.9 Math-comp-analysis

Name: Mathematical Components Analysis

Keyword: Proof assistant

Functional Description: This library adds definitions and theorems for real numbers and their mathematical structures

Release Contributions: Compatible with MathComp 1.12.0, and Coq 8.11, 8.12 and 8.13.

News of the Year: In 2020, we unified norms and absolute values, added a topology and pseudo-metric on the extended reals, and provided a more complete theory about sequences and measure theory.

URL: https://github.com/math-comp/analysis

Publication: hal-01719918

Contact: Cyril Cohen

Participants: Cyril Cohen, Georges Gonthier, Marie Kerjean, Assia Mahboubi, Damien Rouhling, Laurence Rideau, Pierre-Yves Strub, Reynald Affeldt

Partners: Ecole Polytechnique, AIST Tsukuba

6 New results

6.1 Parallel Mask Refreshing Algorithms

Participants Benjamin Grégoire, Gilles Barthe *(IMDEA,MPI-SP)*, Sonia Belaïd *(CryptoExpert)*, François Dupressoir *(University of Surrey)*, Pierre-Alain Fouque *(Université Rennes 1)*, François-Xavier Standaert *(UCL)*, Pierre-Yves Strub *(Ecole Polytechnique)*.

Masking algorithms are solutions for cryptographic engineering when the attacker has access to the hardware.

Refreshing algorithms are a critical ingredient for secure masking. They are instrumental in enabling sound composability properties for complex circuits.

In the paper [4], we improve a proposal of mask refreshing algorithms from EUROCRYPT 2017. First, we provide a generic proof that this algorithm is secure at arbitrary orders—a problem that was left open so far. Second, we use automated tools to further explore the design space of such algorithms and provide the best known parallel mask refreshing gadgets for concretely relevant security orders. Incidentally, we also prove the security of a recent proposal of mask refreshing with improved resistance against horizontal attacks from CHES 2017.

6.2 Private circuits

Participants Benjamin Grégoire, Gaëtan Cassiers *(UCL)*, Itamar Lévi *(UCL)*, François-Xavier Standaert *(UCL)*.

In the paper [6], we focus on hardware implementation of masking algorithms. We first extend the simulability framework and prove that a compositional strategy that is correct without glitches remains valid with glitches. We then use this extended framework to prove the first masked gadgets that enable trivial composition with glitches at arbitrary orders. We show that the resulting "Hardware Private Circuits" approach the implementation efficiency of previously existing but flawed schemes.

6.3 A domain specific language for mask implementations

Participants Benjamin Grégoire, Gilles Barthe *(IMDEA,MPI-SP)*, Marc Gourjon *(Hamburg University of Technology,NXP)*, Maximilian Orlt *(TU Darmstadt)*, Clara Paglialonga *(TU Darmstadt)*, Lars Porth *(TU Darmstadt)*.

We have developed a new approach for building efficient, provably secure, and practically hardened implementations of masked algorithms in assembly language. Our approach is based on a Domain Specific Language in which users can write efficient assembly implementations and fine-grained leakage models (that have to be designed and validated for each specific processor). The latter are then used as a basis for formal verification. The practical benefits of our approach are demonstrated through a case study of the PRESENT S-Box. Our approach significantly narrows the gap between formal verification of masking and practical security. Un article a été soumis pour publication.

6.4 High-assurance high-speed cryptographic implementations

Participants Benjamin Grégoire, José Bacelar Almeida *(INESC TEC)*, Manuel Barbosa *(INESC TEC)*, Gilles Barthe *(IMDEA, MPI-SP)*, Adrien Koutsos *(LSV)*, Vincent Laporte *(Inria, Pesto)*, Tiago Oliveira *(INESC TEC)*, Pierre-Yves Strub *(Ecole Polytechnique)*.

In the paper [9], we have developped a new approach for building cryptographic implementations. Our approach goes the last mile and delivers assembly code that is provably functionally correct, protected against side-channels, and as efficient as hand-written assembly. We illustrate our approach using ChaCha20- Poly1305, one of the two ciphersuites recommended in TLS 1.3, and deliver formally verified vectorized implementations which outperform the fastest non-verified code. This approach combines the Jasmin framework and the EasyCrypt proof assistant.

6.5 Removing inlining in the Jasmin compiler

Participants Benjamin Grégoire, Jean-Christophe Léchenet.

The Jasmin language, presented in [9], has some important limitations. The major one is that all internal functions are systematically inlined by the compiler, this can lead to very large assembly code and can be untrackable for large implementations like the Kyber post-quantum encryption scheme. We have started to develop a new version of the Jasmin language and compiler, that removes this limitation. This new version includes both a new implementation and a new formal verification, with some parts that need to be completely rewritten. This is a work in progress by Jean-Christophe Léchenet.

6.6 Constant-time preserving compilers

Participants Benjamin Grégoire, Swarn Priya, Gilles Barthe *(IMDEA, MPI-SP)*, Rémi Hutin *(Inria, Celtique)*, Vincent Laporte *(Inria, Pesto)*, David Pichardie *(ENS Rennes)*, Alix Trieu *(Aarhus University)*.

In the paper [5], we focus on compilation of cryptographic constant-time programs, and more specifically on the following question: is the code generated by a realistic compiler for a constant-time source program itself provably constant-time? Surprisingly, we answer the question positively for a mildly modified version of the CompCert compiler, a formally verified and moderately optimizing compiler for

C. Concretely, we modify the CompCert compiler to eliminate sources of potential leakage. Then, we instrument the operational semantics of CompCert intermediate languages so as to be able to capture cryptographic constant-time. Finally, we prove that the modified CompCert compiler preserves constant-time. Our mechanization maximizes reuse of the CompCert correctness proof, through the use of new proof techniques for proving preservation of constant-time. These techniques achieve complementary trade-offs between generality and tractability of proof effort, and are of independent interest.

Swarn Priya is currently working on a new technique to prove similar result on the Jasmin compiler.

6.7 High guarantees in the presence of speculative execution

Participants Benjamin Grégoire, Swarn Priya, Gilles Barthe *(IMDEA, MPI-SP)*, Sunjay Cauligi *(UC San Diego)*, Adrien Koutsos *(LSV)*, Kevin Liao *(MPI-SP, MIT)*, Tiago Oliveira *(INESC TEC)*, Swarn Priya *(Inria and Purdue University)*, Tamara Rezk *(Inria, Indes)*, Peter Schwabe *(MPI-SP)*.

Traditionally, resistance to time side chanel attacks are established under a sequential execution semantics. However, this semantics is inconsistent with the behavior of modern processors that make use of speculative execution to improve performance. This mismatch, combined with the high-profile Spectre-style attacks that exploit speculative execution, naturally casts doubts on the robustness of high-assurance cryptography guarantees. We have shown the benefits of high-assurance cryptography extend to speculative execution, costing only a modest performance overhead. We have built atop the Jasmin verification framework an end-to-end approach for proving properties of cryptographic software under speculative execution, and validate our approach experimentally with efficient, functionally correct assembly implementations of ChaCha20 and Poly1305, which are secure against both traditional timing and speculative execution attacks.

6.8 An analysis extension of mathematical components

Participants Cyril Cohen, Reynald Affeldt *(AIST, Japan)*, Marie Kerjean *(Inria Rennes, Gallinette)*, Damien Rouhling *(Inria Nancy, Camus)*, Assia Mahboubi *(Inria Rennes, Gallinette)*, Pierre-Yves Strub *(Ecole Polytechnique)*.

During this year, a large part of the work has been focused on adding extended real numbers (with infinite symbols), sequences, series, measure theory and Lebesgue measure construction. This work led to a publication [8]. The maintenance and evolution of this library confirms the need for a tool supporting the management of hierarchies of mathematical structures, which will be fulfilled by our work on Hierarchy Builder, described in another section.

6.9 Formalization of Spectral theory for matrices

Participant Cyril Cohen.

We are adding to the mathematical components library different elements of matrix theory, especially concerned with diagonalization and trigonalization.

6.10 Formalization of Bourbaki foundations

Participant Laurent Théry.

We have worked on the dissemination of the late José Grimm's work on formalizing set theory as presented in Bourbaki. This is now a library distributed under the name gaia by the coq-community collaborative effort https://github.com/coq-community/gaia.

6.11 Approximations using Chebyshev polynomials

Participants Laurent Théry, Florian Steinberg *(Inria Saclay, Toccata)*.

Florian Steinberg and Laurent Théry have been working on polynomial approximations using Chebyshev polynomials. Extensions for this year make that we can now compute approximations of integrals using Chebyshev models with Coq. This works is available as a library at https://github.com/FlorianSteinberg/Cheby.

6.12 Mathematical Formalization: Tower of Hanoi

Participant Laurent Théry.

The Tower of Hanoi is a puzzle that is linked to some very interesting mathematics. We have formalized various recent resuts about it using the Mathematical Component library, particularly using finite functions and some basic notion of graph theory. This is described in a paper available in a preprint archive [17].

6.13 Formal study of Double-word arithmetic algorithms

Participants Laurence Rideau, Jean-Michel Muller *(CNRS, ENS de Lyon)*.

We finished the formalization of double-word arithmetic algorithms, described in the article *Tight and rigourous error bounds for basic building blocks of double-word arithmetic* [18].

An article describing this work of formalisation has been written and is submitted for publication [16].

The collaboration continues on the formal proof of an algorithm to compute euclidean norms.

6.14 Private types in Elpi

Participants Enrico Tassi, Marco Maggesi *(University of Florence, Italy)*.

When developing large or critical software, the programmer often needs to hide the actual definition of a type and to enforce invariants. OCaml provides an elegant solution in the form of*private types*. This makes it possible to force users to rely on a specific API to construct new pieces of data, but still let them use the `match-with` linguistic construct. We started an investigation about the introduction of this concept in the type system of λ-Prolog. This was described in [13].

6.15 Document management for the Coq system

Participants Enrico Tassi, Maxime Dénès.

We have be re-designing the communication protocol between Coq and its user-interface software to make it compliant with the LSP protocol used in Visual Studio Code.

6.16 Formal proofs on session types

Participants Enrico Tassi, Cinzia Di Giusto *(University of Nice)*, Marco Giunti *(New University of Lisbon)*, Kirstin Peters *(University of Darmstadt)*, Antonio Ravara *(New University of Lisbon)*.

The work on formalizing linear monadic π-calculus that was initiated the previous year has continued. A talk on this topic was given at the VEST workshop.

6.17 Hierarchy Builder

Participants Cyril Cohen, Enrico Tassi, Kazuhiko Sakaguchi *(University of Tsukuba)*.

Building algebraic hierarchies in a proof assistant such as Coq requires a lot of manual labor and often a deep expertise. To reduce the cost, we developed HB, a high level language to build hierarchies of algebraic structures and to make these hierarchies evolve without breaking user code. This relies on Elpi. A paper on this effort was published in an international conference [10]. Presentations about this work were also given at the Coq Workshop 2020 and at the FOMM / Lean Together workshop http://www.andrew.cmu.edu/user/avigad/meetings/fomm2020.

6.18 A guide to use Coq for security evaluations

Participants Maxime Dénès, Yves Bertot, Vincent Laporte *(Inria Rennes, PESTO)*, Arnaud Fontaine *(ANSSI)*, Thomas Letan *(ANSSI)*.

The ANSSI and Inria have been collaborating on guidelines and rules for formal analyses supported by Coq, in order to make these developments easier to read and evaluate in the context of Common Criteria security evaluations. The final document was published by ANSSI in September 2020 https://www.ssi.gouv.fr/uploads/2014/11/anssi-requirements-on-the-use-of-coq-in-the-context-of-common-criteria-evaluations-v1.0-en.pdf.

6.19 Security analysis of ElGamal implementations

Participants Mohamad El Laz, Benjamin Grégoire, Tamara Rezk.

Our study of 26 implementations the ElGamal encryption schemes, showing that 20 of them are insecure because they fail to respect the Decisional Diffie-Hellman assumption led to an article published in an international conference [12].

6.20 Simplification of a constructive version of Tarski's system of geometry

Participants Pierre Boutry.

This is a followup of work on the same topic in the previous year. The axioms have been changed to capture geometry in higher dimension and points that are asserted to exist are unique and depend continuously on parameters, following ideas of Michael Beeson. Future work is concerned with the independence of the new axioms.

6.21 Formalization of the Poincaré disk model in Isabelle

Participants Pierre Boutry, Danijela Simić *(University of Belgrade)*, Filip Marić *(University of Belgrade)*.

The Poincaré disk model is a model that can be shown to satisfy all axioms of Tarski's system of geometry at the exception of the parallel postulate. We developed a formal proof of this fact in the Isabelle system and a paper describing this work was published in a journal [7].

6.22 Propagation of Uncertainty

Participants Pierre Boutry, Vincent Nimal *(Silexica)*, Nestor Demeure *(CEA-DAM)*, Pierre Dossantos-Uzarralde *(CEA-DAM)*, Thomas Mazurkiewicz *(Matsena)*.

An uncertainty can be associated to each constant used in a physical model and a stochastic solution can be computed. The moments of the solution then make it possible to evaluate, e.g., the sensitivity of the solution to some of the model's constants. We have shown that this approach can be applied to Schrödinger's equation. This is particularly interesting because the extra computations can be inserted in numerical algorithms with the help of overloaded operators and types.

6.23 Mutual interpretability of cartesian planes with Tarski's system of geometry

Participants Pierre Boutry, Cyril Cohen, Stéphane Kastenbaum *(ENSIIE Strasbourg)*.

On previous years, we worked on connecting Pierre Boutry's work on axiom systems for geometry and Cyril Cohen's work on quantifier elimination in real closed fields. This year, we extend our work on this topic with a proof that Klein's model satisfies all Tarski axioms but the parallel postulate, whose negation is actually satisfied. This may constitute the first formalized proof of independence of the parallel postulate in Coq.

6.24 Integration of the GeoCoq library to Logipedia

Participants Pierre Boutry, Gaspard Ferey *(Inria Saclay Ile de France, Deducteam project team)*, Julien Narboux *(University of Strasbourg)*, Predrag Janičić *(University of Belgrade)*.

We wish to translate proofs already done in Coq, namely the GeoCoq library, into proofs verifiable by the Dedukti proof system. This requires handling tactics based on internal computation (reflective tactics), that we used intensively in our Coq proof. However, handling reflective tactics is currently not well supported by Dedukti.

We identified the Larus prover `https://github.com/janicicpredrag/Larus` as a way to produce proofs not relying on internal computation. We developed ways to use this prover to handle the concepts of pseudo transitivity and collinearity correctly, but coplanarity still requires attention.

6.25 A Variant of Wagner's Theorem Based on Combinatorial Hypermaps

Participant Christian Doczkal.

Building on the graph theory library we started developing together with Damien Pous [11], we formalized in Coq the main direction of Wagner's theorem, i.e., that every graph without K_5 and $K_{3,3}$ minor is planar. In this work, plane embeddings are represented using combinatorial hypermaps as used in the formal proof of the four-color theorem. This establishes the link between two very different representations of graphs and allowed us to lift the four- color theorem for planar hypermaps to a more standard representation of graphs and the excluded-minor characterization of planarity. All proofs are available as part of the coq-community project https://coq-community.org/graph-theory/wagner/.

6.26 Analysis of pandemic data

Participants Maxime Dénès.

Maxime Dénès volunteered work to the ICUBAM (Intensive Care Unit Bed Availability Monitoring) system and the TousAntiCovid (contact tracing) development efforts. Apart from the practical benefits in fighting against the COVID pandemic, this also led to a publication on statistical models for ICU bed availability [15].

6.27 Vertical cell decomposition for motion planning algorithms

Participants Yves Bertot, Come Neyrand *(ENS Rennes)*.

We developed a formal description for a known algorithm to decompose a plane region into cells that are guaranteed to be free of obstacles. The goal of this development is to make it possible to prove that some trajectories are collision free.

6.28 Formal verification of C programs with floating point computation

Participants Yves Bertot, Andrew Appel *(University of Princeton)*.

We performed a case study of formal verification for a C program involving the semi-naive computation of square roots for floating point numbers, using only elementary operations as in Newton's algorithm. This experiment highlights the possibility to decompose the problem in several parts: C semantics, mathematical view of the algorithm, and floating point operations. This experiment leverages the Coq system, the VST toolkit, the Gappa automatic proof tool, and the Flocq Coq library. An article describing this experiment is published in a journal [3].

7 Bilateral contracts and grants with industry

7.1 Bilateral contracts with industry

The STAMP team participates with the Grace team (Inria Saclay) in the JASMIN contract funded in the framework of the Inria-Nomadic Labs collaboration for research related to the Tezos blockchain. This contract funds the PhD thesis of Swarn Priya.

8 Partnerships and cooperations

8.1 International initiatives

8.1.1 Inria associate team not involved in an IIL

FLAVOR

Title: Formal Library of Analysis for the Verification of Robots

Duration: April 2020 - March 2023

Coordinator: Yves Bertot

Partners: Cyber Physical Security Research Center, AIST (Japan)

Inria contact: Yves Bertot

Summary: The objective is to apply formal methods based on Coq to software and designs that are concerned with robots. Covered topics concern mathematical formalization for real analysis, control theory, kinematic chains, and motion planning.

8.2 International research visitors

8.2.1 Visits of international scientists

Reynald Affeldt of AIST visited our team from January 1st to September 30th.
Swarn Priya visited our team for Master internship while a student at Purdue University.

8.3 National initiatives

8.3.1 ANR

- TECAP "Analyse de protocoles, Unir les outils existants", starting on October 1st, 2017, for 60 months, with a grant of 89 kEuros. Other partners are Inria teams PESTO (Inria Nancy grand-est), Ecole Polytechnique, ENS Cachan, IRISA Rennes, and CNRS. The corresponding researcher for this contract is Benjamin Grégoire.
- SafeTLS "La sécurisation de l'Internet du futur avec TLS 1.3" started on October 1st, 2016, for 60 months, with a grant of 147kEuros. Other partners are Université de Rennes 1, and secrétariat Général de la Défense et de la Sécurité Nationale. The corresponding researcher for this contract is Benjamin Grégoire.
- Scrypt "Compilation sécurisée de primitives cryptographiques" started on February 1st, 2019, for 48 months, with a grant of 100 kEuros. Other partners are Inria team Celtique (Inria Rennes Bretagne Atlantique), Ecole polytechnique, and AMOSSYS SAS. The corresponding researcher for this contract is Benjamin Grégoire.

8.3.2 FUI

The acronym *FUI* stands for "fonds unique interministériel" and is aimed at research and development projects in pre-industrial phase. The STAMP team is part of one such project.

- VERISICC (formal verification for masking techniques for security against side-channel attacks). This contract concerns 5 partners: CRYPTOEXPERTS a company from the Paris region (Île de France), ANSSI (Agence Nationale de Sécurité des Systèmes d'Information), Oberthur Technologies, University of Luxembourg, and STAMP. A sixth company (Ninjalabs) acts as a sub-contractant. The financial grant for STAMP is 391 kEuros, including 111kEuros that are reserved for the sub-contractant. This project started in October 2018 for a duration of 4 years. The corresponding researcher for this contract is Benjamin Grégoire.

9 Dissemination

9.1 Promoting scientific activities

9.1.1 Scientific events: organisation

Member of the organizing committees Yves Bertot is a member of steering committee for the ITP conference.

9.1.2 Scientific events: selection

Member of the conference program committees Enrico Tassi was a member of the PC for WFLP-2020, PADL-2020. Laurent Théry was a member of the programm committee for the Coq Workshop 2020. Cyril Cohen was a member of the PC for CPP 2021. Yves Bertot was a member of the PC for IJCAR2020.

Reviewer Pierre Boutry, Cyril Cohen, and Laurent Théry were reviewers for IJCAR2020. Benjamin Grégoire was a reviewer for AsiaCrypt2020.

9.1.3 Journal

Reviewer - reviewing activities Laurent Théry was a reviewer for JAR (Journal of Automated Reasoning) and LMCS (Logical Methods in Computer Science). Cyril Cohen was a reviewer for MSCS (Mathematical Structures in Computer Science)

9.1.4 Invited talks

Christian Doczkal gave a talk at the 68NQRT seminar in Rennes in October on "Completeness of an Axiomatization of Graph Isomorphism in Coq".

Enrico Tassi gave a talk about Coq-Elpi at University of Saarland https://courses.ps.uni-saarland.de/acp_20/

Enrico Tassi gave a talk about extending Dedukti with Elpi at Deducteam, Inria Saclay Ile de France.

Enrico Tassi gave a talk about the formalization of linear Pi-calculus at the VEST workshop http://groups.inf.ed.ac.uk/abcd/VEST.

Cyril Cohen and Enrico Tassi gave lectures about Coq and Mathcomp at Vrije Universiteit Amsterdam with Assia Mahboubi (EPC Gallinette)

9.1.5 Research administration

Yves Bertot is a member of the steering committee (comité de pilotage) for the Inria-Nomadic Labs collaboration concerning research relevant to the Tezos blockchain.

9.2 Teaching - Supervision - Juries

9.2.1 Teaching

- Master : Yves Bertot, “Proofs and reliable programming using Coq”, 21hours ETD, Sept-Nov 2020, Master Informatique et Interactions, Université Côte d’Azur, France.
- Continuing education : Yves Bertot and Pierre Boutry, "Coq : la preuve par le logiciel", Inria Academy, 28 hours, July, November, December 2020

9.2.2 Supervision

- PhD: Cécile Baritel-Ruet, *Preuves Formelles de la Sécurité de Standards Cryptographiques*, October 2020, supervised by Yves Bertot and Benjamin Grégoire [14].
- Yves Bertot and Laurence Rideau supervise the doctoral thesis of Sophie Bernard.

- Benjamin Grégoire and Tamara Rezk (Indes) supervise the doctoral thesis of Mohamad El Laz.
- Yves Bertot and Benjamin Grégoire supervise the doctoral thesis of Swarn Priya.

9.2.3 Juries

Yves Bertot was a member of the jury for the Habilitation of Assia Mahboubi (Inria Rennes, University of Nantes).

9.3 Popularization

9.3.1 Internal or external Inria responsibilities

Laurence Rideau is a member of the editorial board for Interstices, an Inria sponsored outreach publication on the web.

9.3.2 Animation

Cyril Cohen opened a Zulip chat server, on 2020-05-06, for the Coq community at large, fostering communications way beyond gitter, discourse and coq-club. It regroups streams for several projects beyond Coq itself (CertiCoq, Coq Platform, coq-community, stdlib2, coq-elpi, equations, fiat-crypto, FreeSpec, Hierarchy Builder, hs-to-coq, jsCoq, math-comp and math-comp analysis, MetaCoq, Mtac2, SerAPI, User interfaces and VsCoq), Coq related workshops (Coq workshop 2020 and CUDW 2020 so far) and now schools and tutorials.

- Exported gitter data.
- Co-admin with Théo Zimmermann.
- Statistics https://coq.zulipchat.com/stats:
  - almost 40k messages sent and growing linearly
  - more than 600 users, roughly 50 to 70 daily active users, roughly between 100 and 150 fortnightly active user.
  - 80% messages are sent over public streams, 17% private
  - More than 1.5M message read.

10 Scientific production

10.1 Major publications

[1] R. Affeldt, C. Cohen and D. Rouhling. 'Formalization Techniques for Asymptotic Reasoning in Classical Analysis'. In: *Journal of Formalized Reasoning* (Oct. 2018). URL: https://hal.inria.fr/hal-01719918.

[2] J. B. Almeida, M. Barbosa, G. Barthe, A. Blot, B. Grégoire, V. Laporte, T. Oliveira, H. Pacheco, B. Schmidt and P.-Y. Strub. 'Jasmin: High-Assurance and High-Speed Cryptography'. In: *CCS 2017 - Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Dallas, United States, Oct. 2017, pp. 1–17. URL: https://hal.archives-ouvertes.fr/hal-01649140.

10.2 Publications of the year

International journals

[3] A. W. Appel and Y. Bertot. 'C-language floating point proofs layered with VST and Flocq'. In: *Journal of Formalized Reasoning* 13.1 (21st Dec. 2020), pp. 1–16. DOI: 10.6092/issn.1972-5787/11442. URL: https://hal.inria.fr/hal-03130704.

[4] G. Barthe, S. Belaïd, F. Dupressoir, P.-A. Fouque, B. Grégoire, F.-X. Standaert and P.-Y. Strub. 'Improved parallel mask refreshing algorithms: generic solutions with parametrized non-interference and automated optimizations'. In: *Journal of Cryptographic Engineering* 10.1 (Apr. 2020), pp. 17–26. DOI: 10.1007/s13389-018-00202-2. URL: https://hal.inria.fr/hal-03133221.

[5] G. Barthe, S. Blazy, B. Grégoire, R. Hutin, V. Laporte, D. Pichardie and A. Trieu. 'Formal verification of a constant-time preserving C compiler'. In: *Proceedings of the ACM on Programming Languages* 4.POPL (Jan. 2020), pp. 1–30. DOI: 10.1145/3371075. URL: https://hal.univ-lorraine.fr/hal-02975012.

[6] G. Cassiers, B. Grégoire, I. Levi and F.-X. Standaert. 'Hardware Private Circuits: From Trivial Composition to Full Verification'. In: *IEEE Transactions on Computers* (9th Sept. 2020). DOI: 10.1109/tc.2020.3022979. URL: https://hal.inria.fr/hal-03133227.

[7] D. Simić, F. Marić and P. Boutry. 'Formalization of the Poincaré Disc Model of Hyperbolic Geometry'. In: *Journal of Automated Reasoning* 65.1 (30th Apr. 2020), pp. 31–73. DOI: 10.1007/s10817-020-09551-2. URL: https://hal.inria.fr/hal-03120829.

International peer-reviewed conferences

[8] R. Affeldt, C. Cohen, M. Kerjean, A. Mahboubi, D. Rouhling and K. Sakaguchi. 'Competing inheritance paths in dependent type theory: a case study in functional analysis'. In: IJCAR 2020 - International Joint Conference on Automated Reasoning. Paris, France, 29th June 2020, pp. 1–19. URL: https://hal.inria.fr/hal-02463336.

[9] J. B. Almeida, M. Barbosa, G. Barthe, B. Grégoire, A. Koutsos, V. Laporte, T. Oliveira and P.-Y. Strub. 'The Last Mile: High-Assurance and High-Speed Cryptographic Implementations'. In: SP 2020 - IEEE Symposium on Security and Privacy. 2020 IEEE Symposium on Security and Privacy (SP). San Francisco, United States: https://www.ieee-security.org/TC/SP2020/index.html, May 2020, pp. 965–982. DOI: 10.1109/SP40000.2020.00028. URL: https://hal.univ-lorraine.fr/hal-02974993.

[10] C. Cohen, K. Sakaguchi and E. Tassi. 'Hierarchy Builder: algebraic hierarchies made easy in Coq with Elpi'. In: FSCD 2020 - 5th International Conference on Formal Structures for Computation and Deduction. 5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020) 167. Paris, France, 2020, 34:1–34:21. URL: https://hal.inria.fr/hal-02478907.

[11] C. Doczkal and D. Pous. 'Completeness of an Axiomatization of Graph Isomorphism via Graph Rewriting in Coq'. In: CPP 2020 - 9th ACM SIGPLAN International Conference on Certified Programs and Proofs. Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '20). New Orleans, LA, United States, 2020. DOI: 10.1145/3372885.3373831. URL: https://hal.archives-ouvertes.fr/hal-02333553.

[12] M. El Laz, B. Grégoire and T. Rezk. 'Security Analysis of ElGamal Implementations'. In: SECRYPT 2020 - 17th International Conference on Security and Cryptography. Lieusaint - Paris, France, 8th July 2020, pp. 310–321. DOI: 10.5220/0009817103100321. URL: https://hal.inria.fr/hal-03141511.

Conferences without proceedings

[13] M. Maggesi and E. Tassi. 'Private types in Higher Order Logic Programming'. In: Workshop on Trends, Extensions, Applications and Semantics of Logic Programming (TEASE-LP). Virtual Event, France, 28th May 2020. URL: https://hal.inria.fr/hal-03117762.

Doctoral dissertations and habilitation theses

[14] C. Baritel-Ruet. 'Formal Security Proofs of Cryptographic Standards'. Université côte d'azur, 2nd Oct. 2020. URL: https://tel.archives-ouvertes.fr/tel-03150443.

Reports & preprints

[15] C. ICUBAM, L. Bonnasse-Gahot, M. Dénès, G. Dulac-Arnold, S. Girgin, F. Husson, V. Iovene, J. Josse, A. Kimmoun, F. Landes, J.-P. Nadal, R. Primet, F. Quintao, P. G. Raverdy, V. Rouvreau, O. Teboul and R. Yurchak. *ICU Bed Availability Monitoring and analysis in the Grand Est region of France during the COVID-19 epidemic*. 25th May 2020. URL: https://hal.archives-ouvertes.fr/hal-02620018.

[16] J.-M. Muller and L. Rideau. *Formalization of double-word arithmetic, and comments on "Tight and rigorous error bounds for basic building blocks of double-word arithmetic"*. 20th Oct. 2020. URL: https://hal.archives-ouvertes.fr/hal-02972245.

[17] L. Théry. *Playing with the Tower of Hanoi Formally*. 21st July 2020. URL: https://hal.inria.fr/hal-02903548.

10.3 Cited publications

[18] M. M. Joldes, J.-M. Muller and V. Popescu. 'Tight and rigourous error bounds for basic building blocks of double-word arithmetic'. In: *ACM Transactions on Mathematical Software* 44.2 (2017), pp. 1–27. DOI: 10.1145/3121432. URL: https://hal.archives-ouvertes.fr/hal-01351529.